add2 Enabling Innovation

DTB Training

Unit 9: DTB Vehicle Model - Part B

System I/O Interfacing via CAN Messages

Simon Clarke

Technical Director

Prerequisites

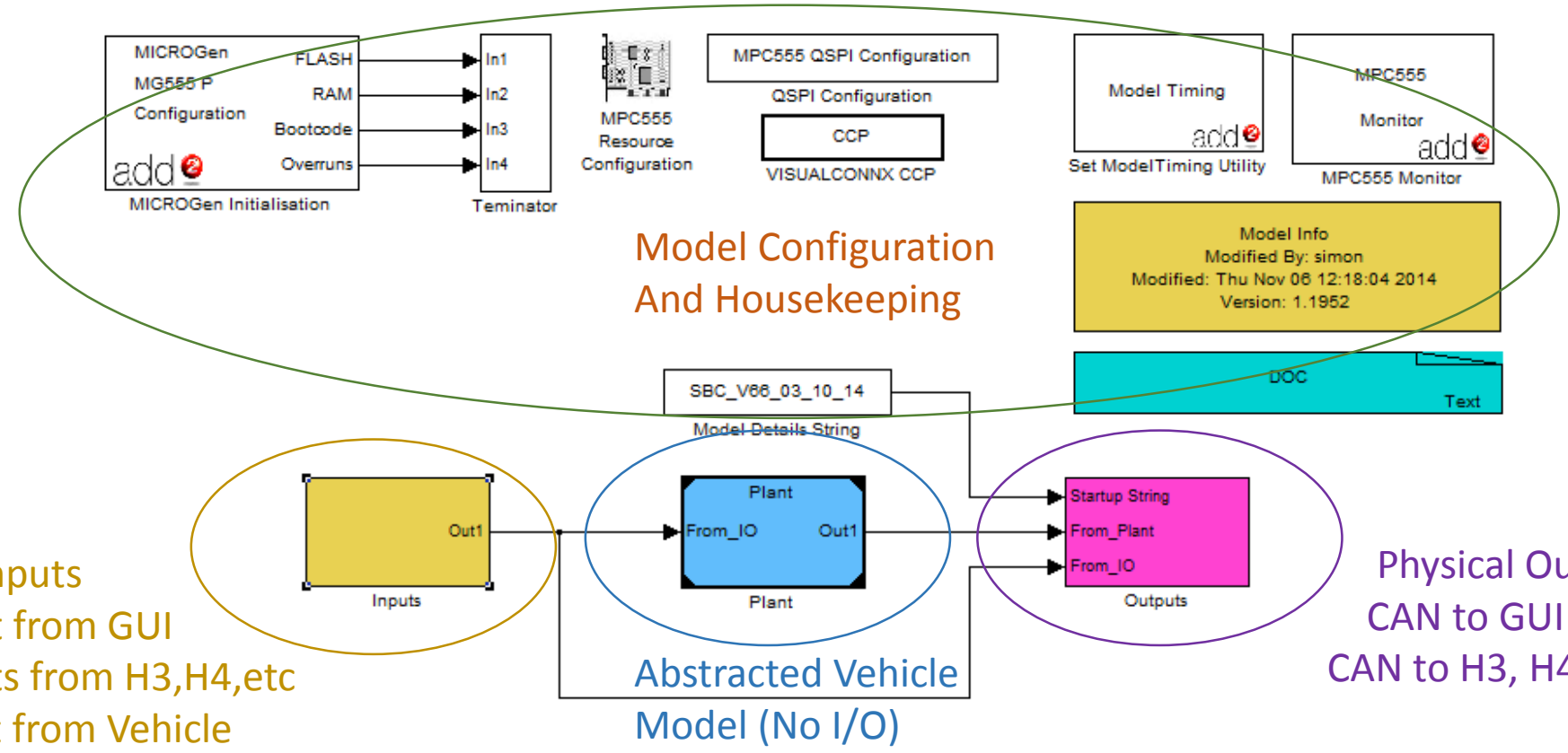
- Knowledge of CAN bus technology
- Knowledge of Matlab, Simulink and Simulink Coder
  (Formerly Real-time-workshop)
- Familiarity with training course units 1-9a
- A fully configured workstation toolchain (for making and viewing model changes)
- Access to the DTB system
- H3-DSP-GUI Installation for changing configurations

Agenda for Unit 9b

- Introduction to vehicle model
  - Vehicle Model Refresher
    - Top level only (Kyaw will do a separate one for engine/transmission/vehicle & Environment)
    - Discuss Sample time colours and associated tips
    - Show off-line simulation and techniques for Simulink Passthrough
- Preparing to make changes to the model
  - How the CAN interface is defined
  - How CAN databases are organised in GUI and Model
  - How to change the VISUALCONNX interface (Key points)
- MICROGen blockset training (V3.0.2)
  - Front Panel, VISUALCONNX, Inputs, Outputs, Additional CAN, RS232, Power Control
  - More detail on the blockset, such as NV Storage, Enhanced Serial, External Display
- Questions and feedback

DTB Vehicle Model – **Vehicle Model Continued**

- The top level of the model is structured as follows:



DTB Vehicle Model –**Plant Model Sub system**

Previous Demo Showed

- Discussed model referencing
- We previously looked at mains parts (Limited time for too much detail)
  - Engine
    - How different engines are brought in
  - Drivetrain
    - How different transmissions are brought in
  - Vehicle and Environment
    - Simple vehicle dynamics and bringing in Vehicle Mass Figures etc.
- More from Kyaw on this later

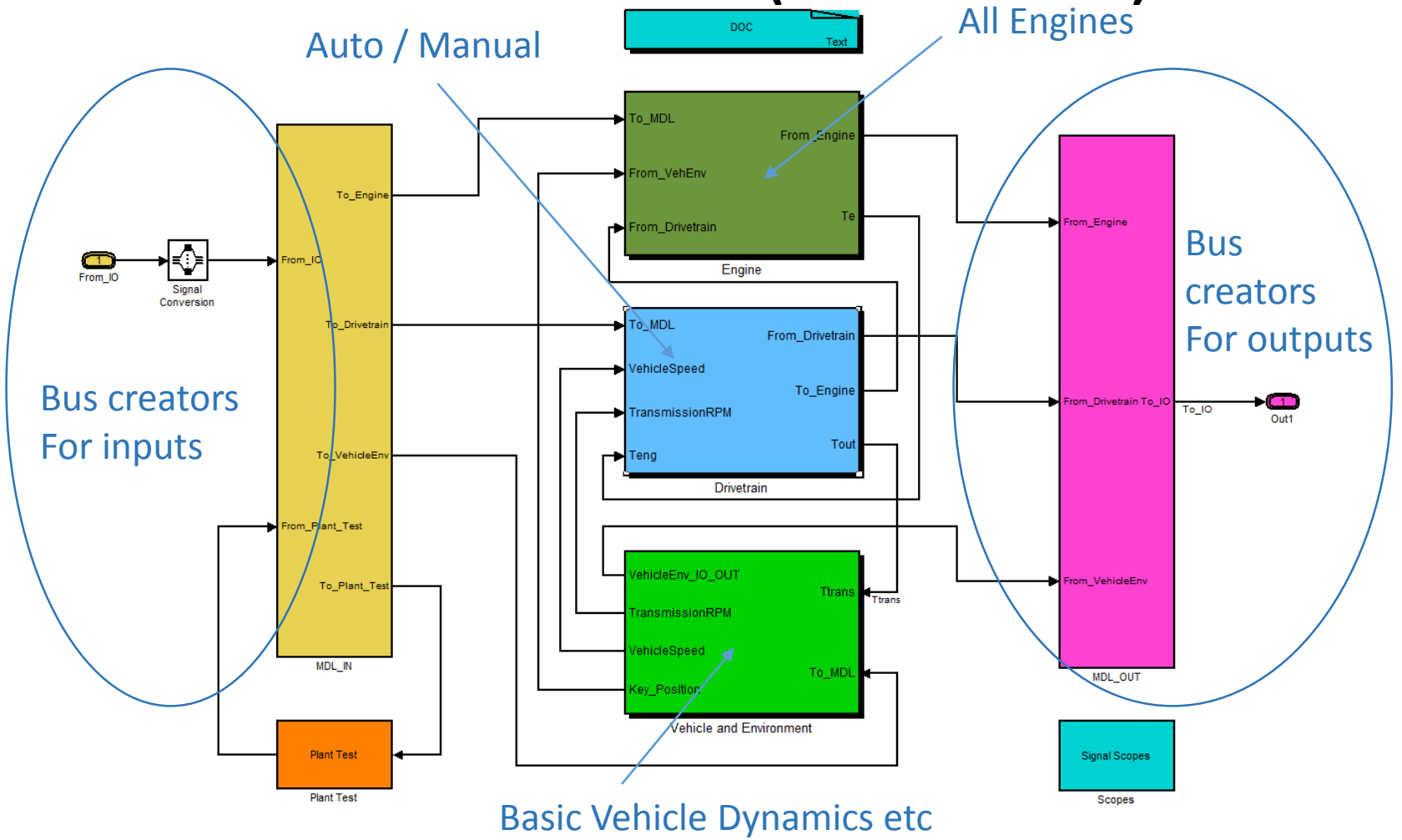
- Discuss debugging via offline simulation
  - Signal Scopes
    - For testing signal behaviour
    - Simulink Passthrough of blocks (Covered later, not to be confused by ‘DTB passthrough’)

DTB Vehicle Model – **Plant Model (referenced)**

The Vehicle simulation is a separate “referenced” model

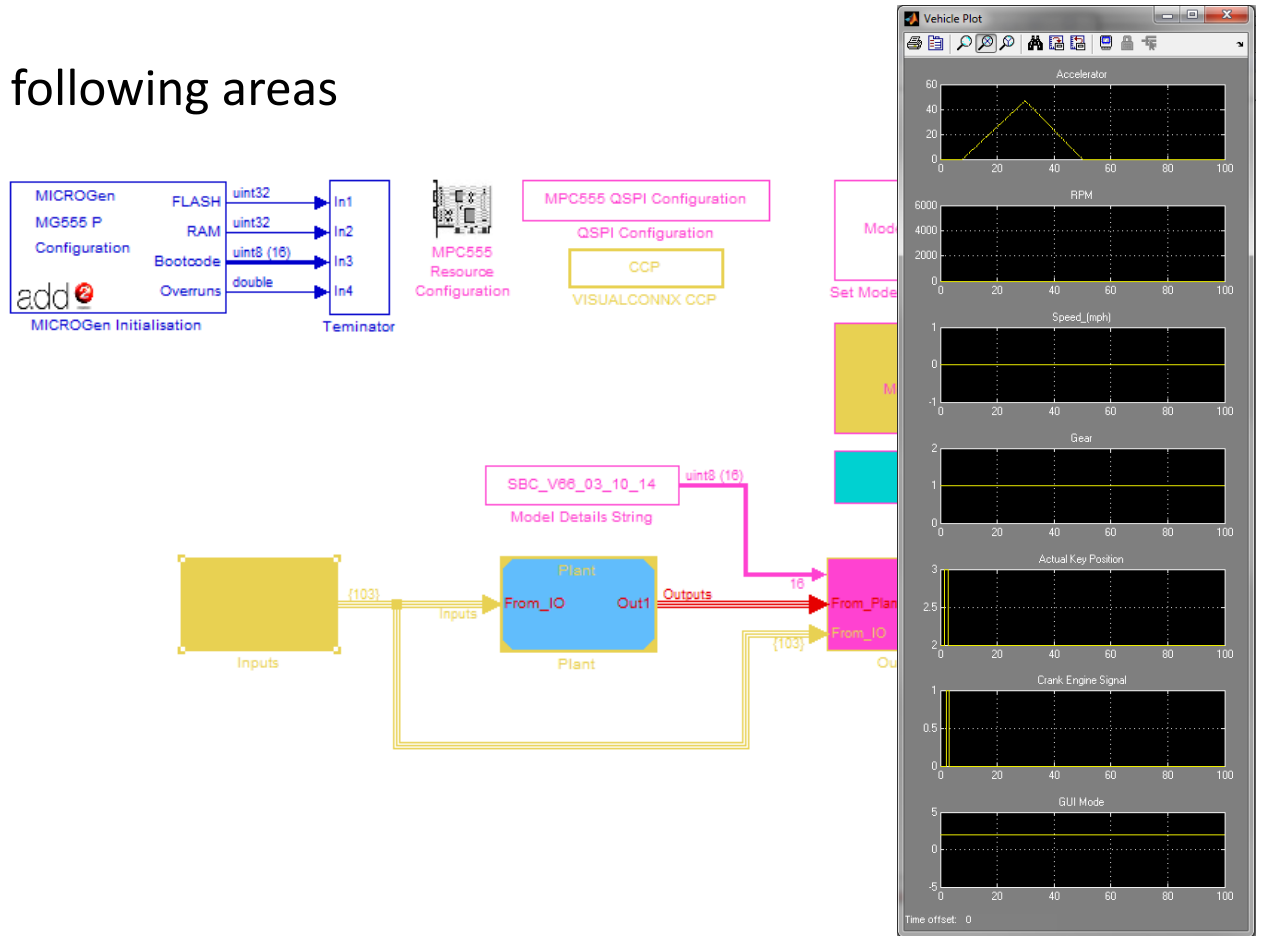
This is loaded by the main model.

Can simulate separately.



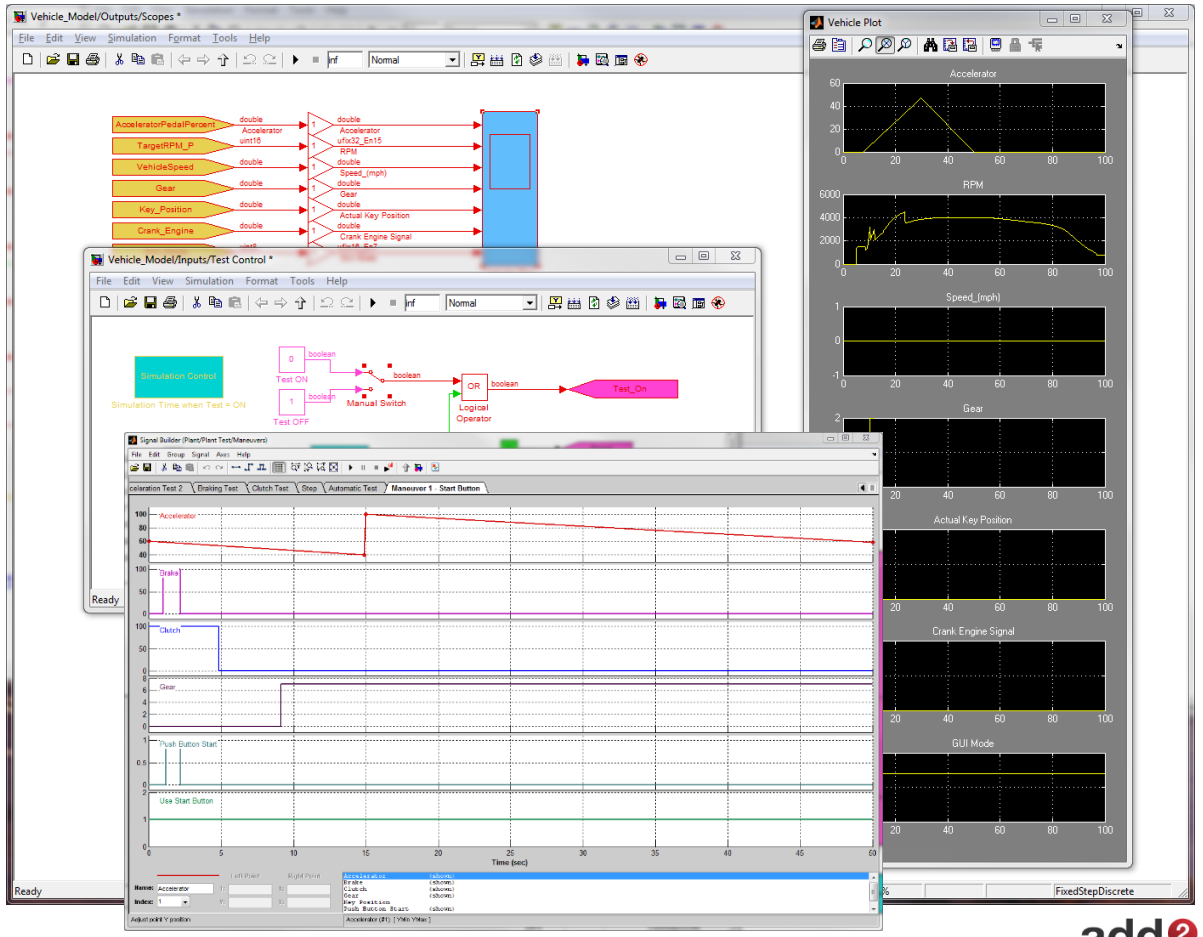
DTB Vehicle Model – **Off-line Simulation from top level**

- A look at the vehicle model, covering the following areas
  - Updating diagram
    - Sample time colours
    - Wide non-scalar lines
    - Port Data Type
    - Signal Dimensions
  - Test Control
    - How to run off-line
    - Looking at the Scopes (O/Ps)
  - Offline Pass-through (Simulation Input)
    - Input Simulation
    - Output Checking
    - Useful for debugging



DTB Vehicle Model – **Off-line Simulation (Continued)**

- The inputs sub system controls the test mode…
  - Use “Manual Switch”
    - Test ON – For Offline Simulation
    - Test OFF – For Real-time Builds
  - Useful to check model is roughly close
  - Can add scopes and other diagnostics
  - Can check look-up tables etc
  - Use Simulation Pass-through
    - Remember Analogue Simulation Pass-through Inputs are normalised to ‘1’
    - Remember Analogue Simulation Outputs are normalised to ‘1’
- Remember to turn Test switch OFF before deploying to MICROGen!

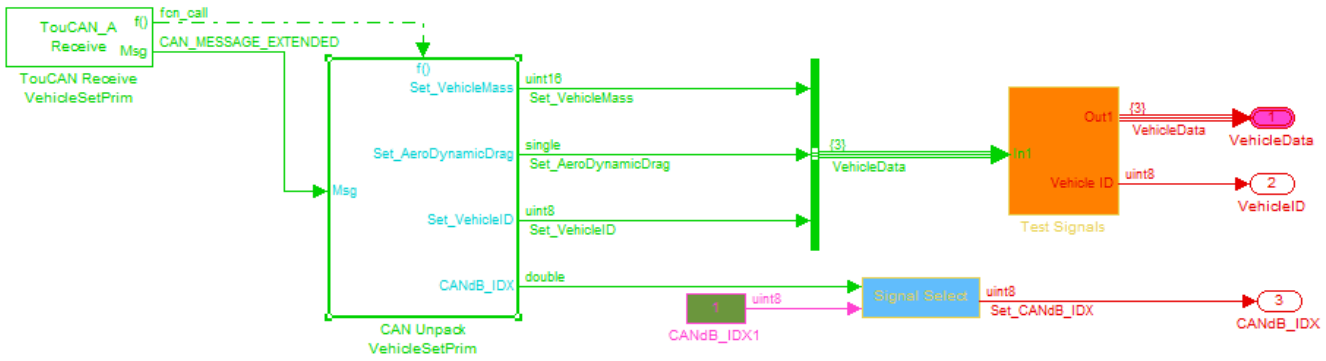


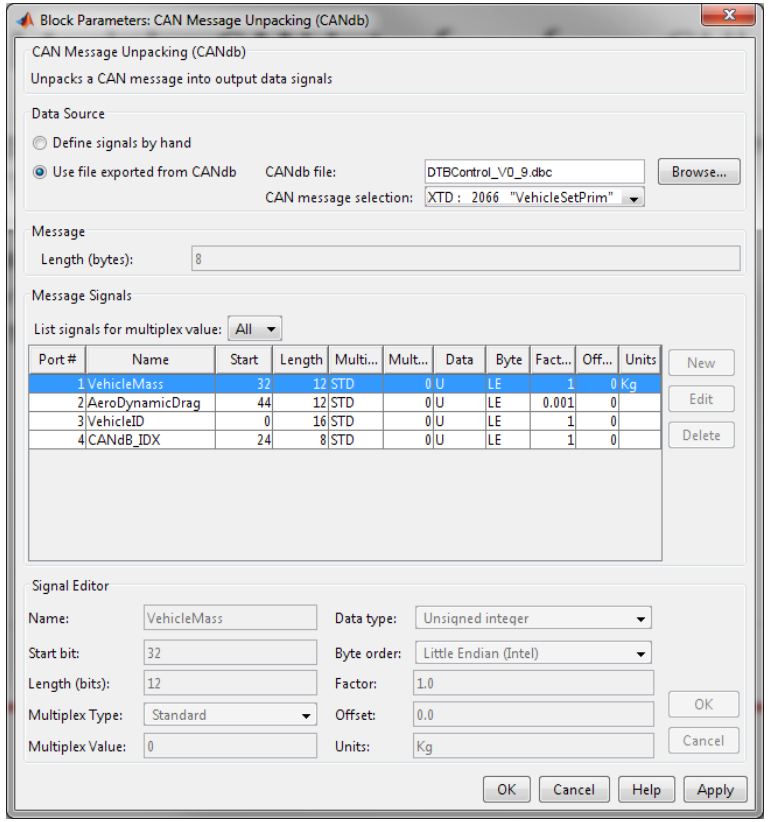
DTB Vehicle Model – **GUI and I/O Interfaces**

- This is an important subject
  - You need to understand this to make any changes to the GUI interface
  - Usually you will be adding signals to CAN messages and updating the model & GUI to match
- A look at the model, covering the following areas
  - How control signals are brought in from the GUI
    - This uses a CANdB currently “DTBControl_V0_91.dbc”
    - Ensure this file is kept versioned
    - Ensure it is re-imported into the GUI
  - How real inputs are brought in from the H3/H4 System
    - “HIL_Genix.System.dbc”
    - This is generated from the hardware configuration H3-DSP-GUI
  - See slide below for further CAN interface information

DTB Vehicle Model – **CAN Interface from GUI**

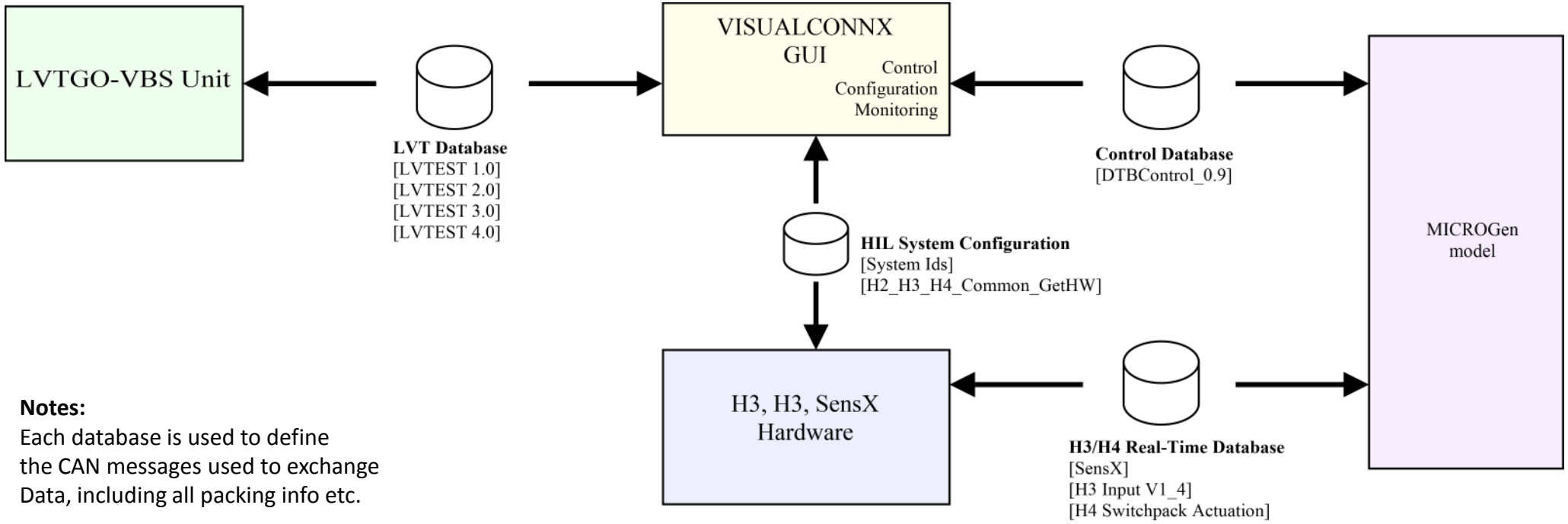
- To exchange messages between the GUI and the model, a CAN database is used.
  - Currently this is “DTBControl_V0_9.dbc”
  - VISUALCONNX needs to load this database along with others
    - VISUALCONNX uses a set of CAN databases to build up its “connection”
    - See later slides for more info
  - The model loads the database into CAN blocks
  - Make sure the Message Selection is correct (especially if changing dbc)




Block Parameters: CAN Message Unpacking (CANdb)
CAN Message Unpacking (CANdb)
Unpacks a CAN message into output data signals
Data Source
Define signals by hand
Use file exported from CANdb
CANdb file: DTBControl_V0_9.dbc
Browse...
CAN message selection: XTD : 2066 "VehicleSetPrim"
Message
Length (bytes): 8
Message Signals
List signals for multiplex value: All
Port # | Name | Start | Length | Multi... | Mult... | Data | Byte | Fact... | Off... | Units
1 VehicleMass 32 12 STD 0 U LE 1 0 Kg
2 AeroDynamicDrag 44 12 STD 0 U LE 0.001 0
3 VehicleID 0 16 STD 0 U LE 1 0
4 CANdB_IDX 24 8 STD 0 U LE 1 0
New
Edit
Delete
Signal Editor
Name: VehicleMass
Data type: Unsigned integer
Start bit: 32
Byte order: Little Endian (Intel)
Length (bits): 12
Factor: 1.0
Multiplex Type: Standard
Offset: 0.0
Multiplex Value: 0
Units: Kg
OK
Cancel
OK
Cancel
Help
Apply

DTB Vehicle Model – **Database usage in DTB**

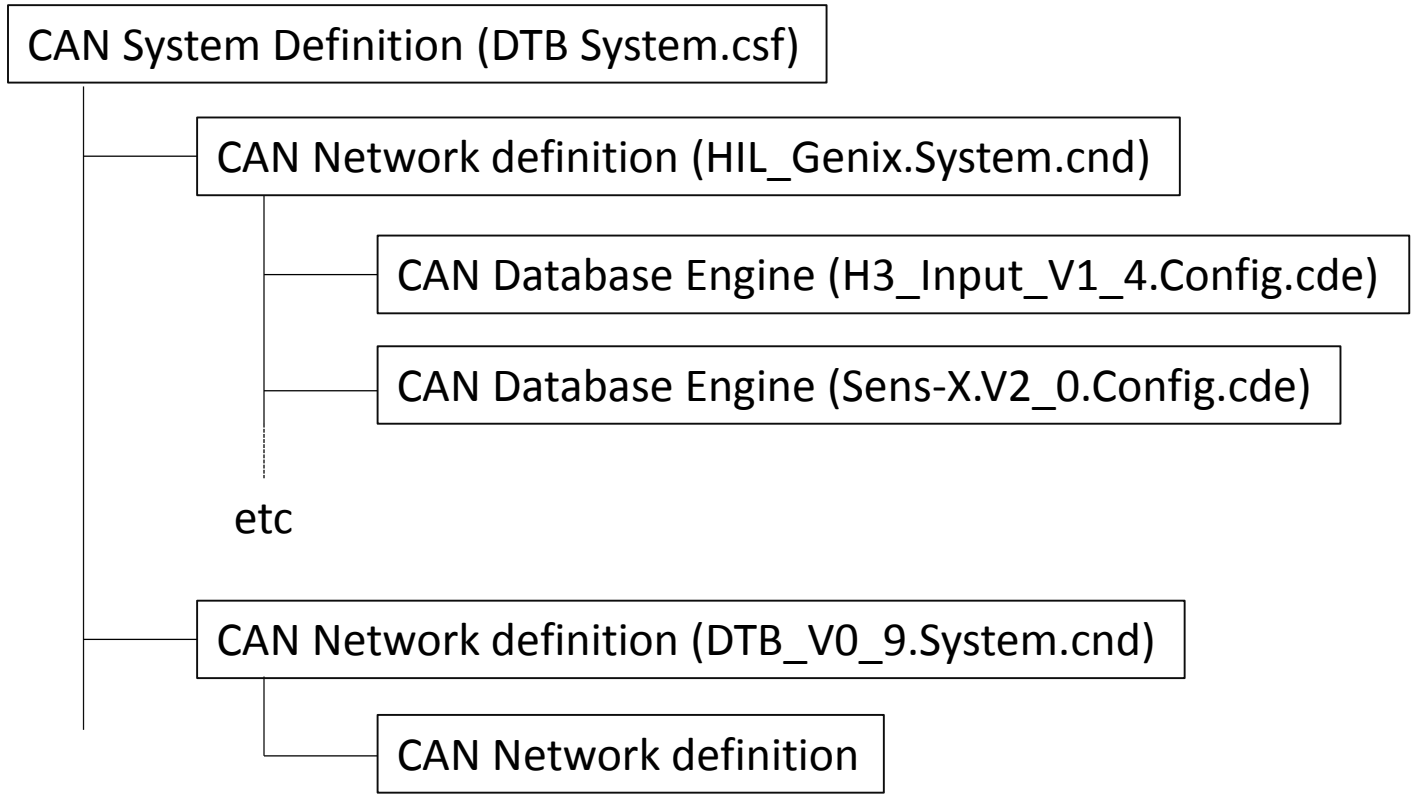
- The following figure shows how each database is used in the DTB system



Notes:
Each database is used to define the CAN messages used to exchange Data, including all packing info etc.

DTB Vehicle Model – **CAN Interface from GUI Continued**

- To load the CAN database into VISUALCONNX, a system network definition file is used.



These files are loaded by VISUALCONNX each time the DTB GUI is loaded.
HIL Genix System should not need to be changed, unless we change it.

DTB_V0_9.System.cnd can be changed and exported to CDE format as required.

DTB Vehicle Model – **CAN System Definition**

- The CAN System Definition is defined in the VISUALCONNX user manual, however you probably only need to change it for version control reasons…

```
[Main]
Networks=2

[Network 1]
File=HIL_Genix.System.cnd

[Network 2]
File=DTB_V0_9.System.cnd
```

- HIL_Genix_System.cnd – Defines the CAN network for all HIL I/O.
- DTB_V0_9.System.cnd – Defines the interface from GUI to Model and related items.

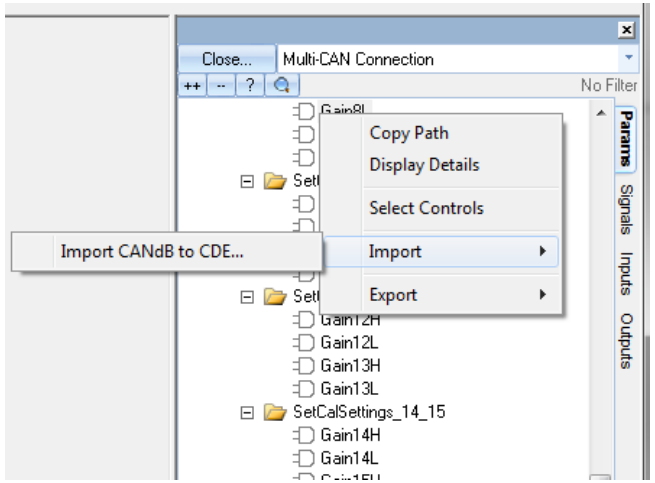
DTB Vehicle Model – **CAN Network Definition**

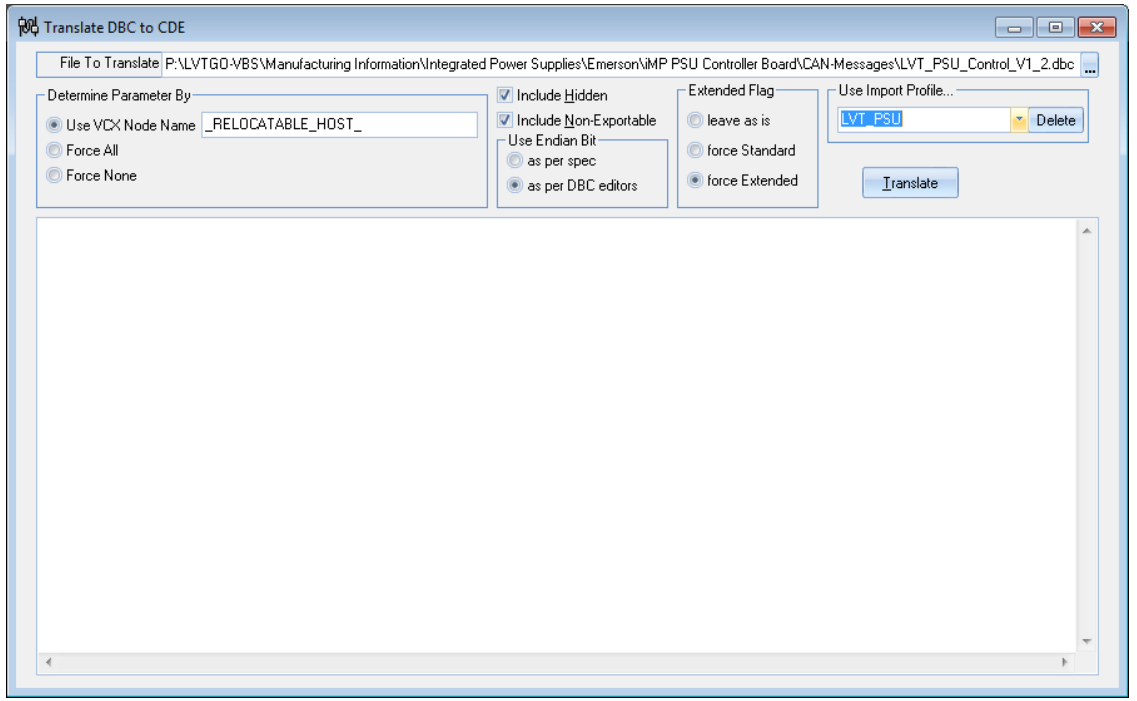
- The CAN System Definition format is defined in the VISUALCONNX user manual.
- Change it for version control (eg: DTBControl_V0_9_1.cde)

```
[Main]
Databases=3
Network=0

[Database 1]
File=DTBControl_V0_9.cde
BaseID=&H0
Groups=0

[Database 2]
File=LV Test SigParamSpec Version4_1_1.cde
BaseID=&H0
Groups=0

[Database 3]
File=H4_Switchpack_Actuation.V1_3_MODIFIED.cde
BaseID=&H0
Groups=0
```

DTB Vehicle Model – **CAN Database Engine**

- The CAN Database Engine format is defined in the VISUALCONNX user manual, however, rather than manually editing this file, you should use a standard database editor and **import** the data into VISUALCONNX developer.

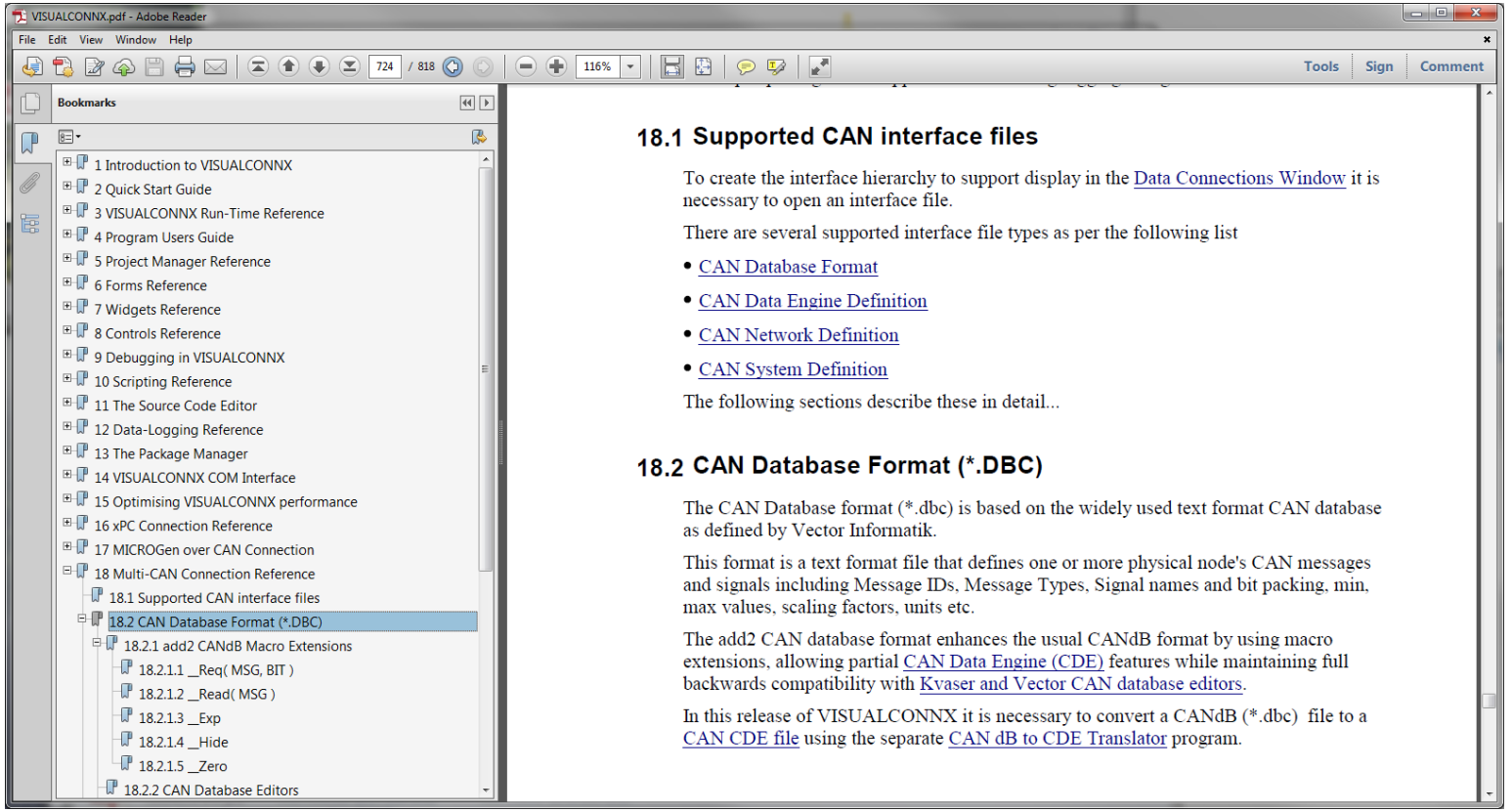




DTB Vehicle Model – **CAN Database Imports**

- For a full reference see:
  - VISUALCONNX User Manual
    - Chapter 18
    - Appendix 3

- However:
  - You don't have to know all this info!
  - You need to just know enough to import the database for each change you make

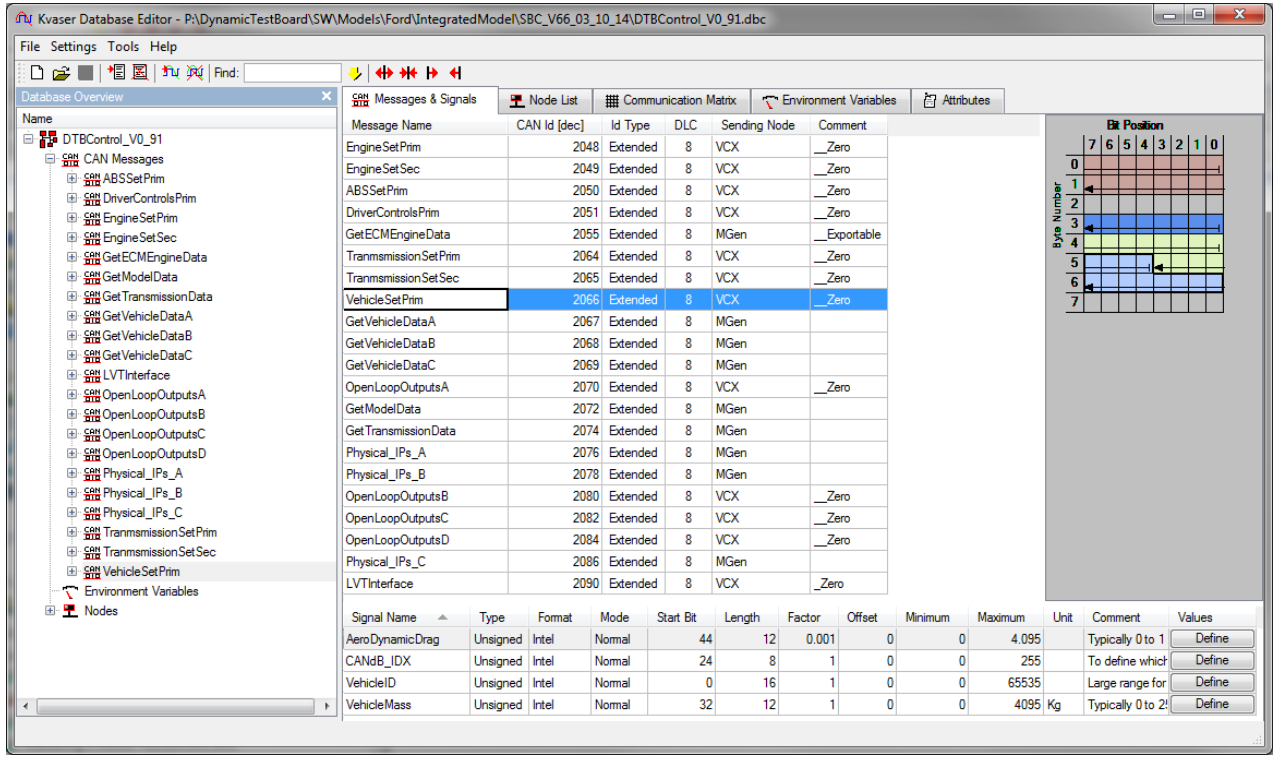

VISUALCONNX.pdf - Adobe Reader
File Edit View Window Help
Tools Sign Comment
Bookmarks
1 Introduction to VISUALCONNX
2 Quick Start Guide
3 VISUALCONNX Run-Time Reference
4 Program Users Guide
5 Project Manager Reference
6 Forms Reference
7 Widgets Reference
8 Controls Reference
9 Debugging in VISUALCONNX
10 Scripting Reference
11 The Source Code Editor
12 Data-Logging Reference
13 The Package Manager
14 VISUALCONNX COM Interface
15 Optimising VISUALCONNX performance
16 xPC Connection Reference
17 MICROGen over CAN Connection
18 Multi-CAN Connection Reference
18.1 Supported CAN interface files
18.2 CAN Database Format (*.DBC)
18.2.1 add2 CANdB Macro Extensions
18.2.1.1 _Req(MSG, BIT)
18.2.1.2 _Read(MSG)
18.2.1.3 _Exp
18.2.1.4 _Hide
18.2.1.5 _Zero
18.2.2 CAN Database Editors
18.1 Supported CAN interface files
To create the interface hierarchy to support display in the Data Connections Window it is necessary to open an interface file.
There are several supported interface file types as per the following list
CAN Database Format
CAN Data Engine Definition
CAN Network Definition
CAN System Definition
The following sections describe these in detail...
18.2 CAN Database Format (*.DBC)
The CAN Database format (*.dbc) is based on the widely used text format CAN database as defined by Vector Informatik.
This format is a text format file that defines one or more physical node's CAN messages and signals including Message IDs, Message Types, Signal names and bit packing, min, max values, scaling factors, units etc.
The add2 CAN database format enhances the usual CANdB format by using macro extensions, allowing partial CAN Data Engine (CDE) features while maintaining full backwards compatibility with Kvaser and Vector CAN database editors.
In this release of VISUALCONNX it is necessary to convert a CANdB (*.dbc) file to a CAN CDE file using the separate CAN dB to CDE Translator program.

DTB Vehicle Model – **CAN Interface from GUI Continued**

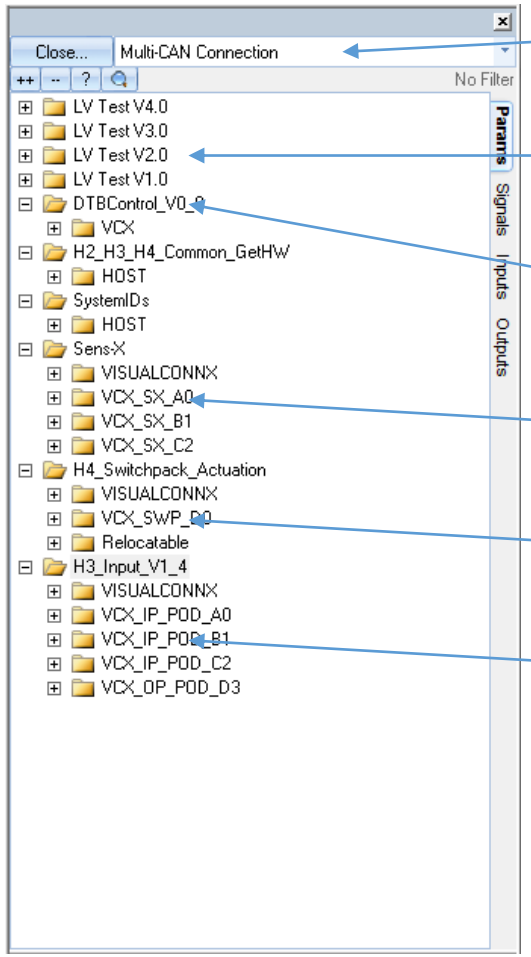
- To view or edit the DTB Control CAN database use a tool such as Kvaser's CAN database editor or notepad ☺

- This allows you to edit or create new signals or parameters for the MICROGen model, such as adding new sensors, readbacks etc

- This database editor is freely available from Kvaser's website or you can use the Vector tool if available.


Kvaser Database Editor - P:\DynamicTestBoard\SW\Models\Ford\IntegratedModel\SBC_V66_03_10_14\DTBControl_V0_91.dbc
File Settings Tools Help
Find:
Database Overview
Name
DTBControl_V0_91
CAN Messages
ABSSetPrim
DriverControlsPrim
EngineSetPrim
EngineSetSec
GetECMEngineData
GetModelData
GetTransmissionData
GetVehicleDataA
GetVehicleDataB
GetVehicleDataC
LVTInterface
OpenLoopOutputsA
OpenLoopOutputsB
OpenLoopOutputsC
OpenLoopOutputsD
Physical_IPs_A
Physical_IPs_B
Physical_IPs_C
TranmsmissionSetPrim
TranmsmissionSetSec
VehicleSetPrim
Environment Variables
Nodes
Messages & Signals
Node List
Communication Matrix
Environment Variables
Attributes
Message Name | CAN Id [dec] | Id Type | DLC | Sending Node | Comment
EngineSetPrim | 2048 | Extended | 8 | VCX | _Zero
EngineSetSec | 2049 | Extended | 8 | VCX | _Zero
ABSSetPrim | 2050 | Extended | 8 | VCX | _Zero
DriverControlsPrim | 2051 | Extended | 8 | VCX | _Zero
GetECMEngineData | 2055 | Extended | 8 | MGen | _Exportable
TranmsmissionSetPrim | 2064 | Extended | 8 | VCX | _Zero
TranmsmissionSetSec | 2065 | Extended | 8 | VCX | _Zero
VehicleSetPrim | 2066 | Extended | 8 | VCX | _Zero
GetVehicleDataA | 2067 | Extended | 8 | MGen
GetVehicleDataB | 2068 | Extended | 8 | MGen
GetVehicleDataC | 2069 | Extended | 8 | MGen
OpenLoopOutputsA | 2070 | Extended | 8 | VCX | _Zero
GetModelData | 2072 | Extended | 8 | MGen
GetTransmissionData | 2074 | Extended | 8 | MGen
Physical_IPs_A | 2076 | Extended | 8 | MGen
Physical_IPs_B | 2078 | Extended | 8 | MGen
OpenLoopOutputsB | 2080 | Extended | 8 | VCX | _Zero
OpenLoopOutputsC | 2082 | Extended | 8 | VCX | _Zero
OpenLoopOutputsD | 2084 | Extended | 8 | VCX | _Zero
Physical_IPs_C | 2086 | Extended | 8 | MGen
LVTInterface | 2090 | Extended | 8 | VCX | _Zero
Bit Position
7 6 5 4 3 2 1 0
Byte Number
0 1 2 3 4 5 6 7
Signal Name | Type | Format | Mode | Start Bit | Length | Factor | Offset | Minimum | Maximum | Unit | Comment | Values
AeroDynamicDrag | Unsigned | Intel | Normal | 44 | 12 | 0.001 | 0 | 0 | 4.095 | | Typically 0 to 1 | Define
CANdB_IDX | Unsigned | Intel | Normal | 24 | 8 | 1 | 0 | 0 | 255 | | To define which | Define
VehicleID | Unsigned | Intel | Normal | 0 | 16 | 1 | 0 | 0 | 65535 | | Large range for | Define
VehicleMass | Unsigned | Intel | Normal | 32 | 12 | 1 | 0 | 0 | 4095 | Kg | Typically 0 to 2 | Define

DTB Vehicle Model – **GUI's Hierarchical View of Databases**



Databases are "loaded" into VISUALCONNX's "Multi-CAN" connection

LVTest CAN Databases

DTB GUI Interface Database
Imported from DTBControl_V0_9

SensX Fuse Interfaces

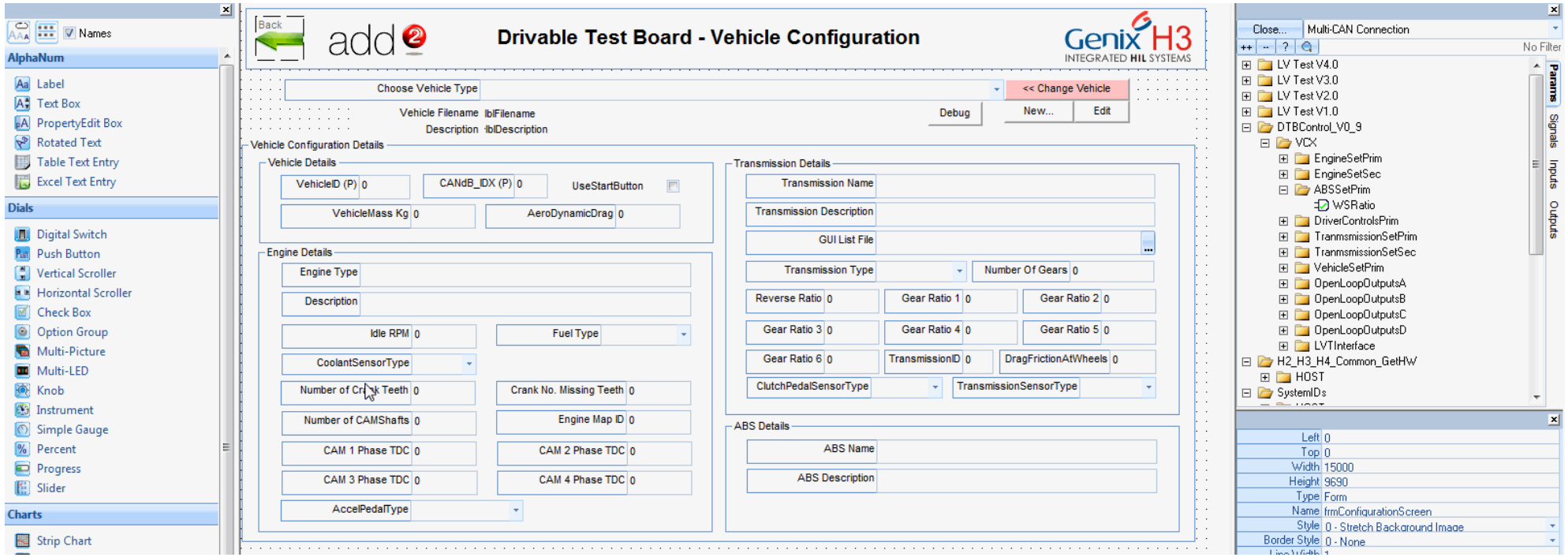
H4 Resistance Module Interface

H3 DSP Module Interface

VISUALCONNX shows this hierarchy in the "tree" when in developer mode.

DTB Vehicle Model – **Linking CAN signals in GUI**

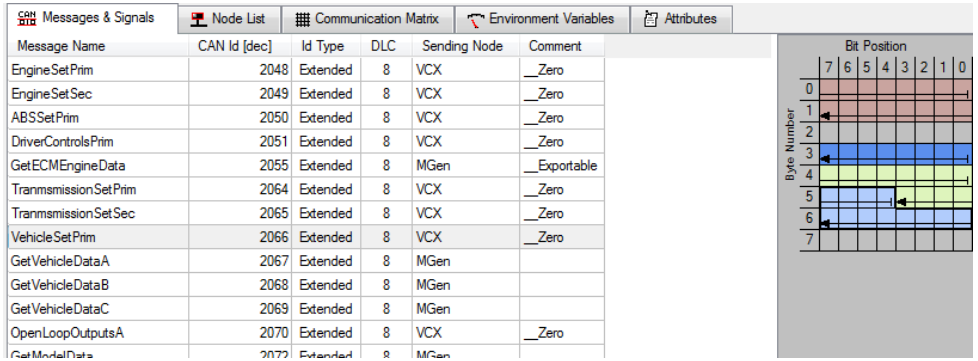
- It is the act of “dragging” the node in the VISUALCONNX CAN database tree that links the CAN message to the GUI control you drop it on – Watch this in action…



We will cover how this is done in more detail with the VISUALCONNX training unit

DTB Vehicle Model – CAN Interface from GUI Continued

- Parameters vs Signals
  - A ‘parameter’ is a value that is sent from VISUALCONNX to make a change to something in the model
  - A ‘signal‘ is a value that is sent from the model to the GUI to be displayed, recorded etc
  - To tell the CAN Database which direction the CAN message goes we set the “node” name in the database
    - MGen is MICROGen
    - VCX is VISUALCONNX


Messages & Signals
Node List
Communication Matrix
Environment Variables
Attributes
Message Name | CAN Id [dec] | Id Type | DLC | Sending Node | Comment
EngineSetPrim | 2048 | Extended | 8 | VCX | _Zero
EngineSetSec | 2049 | Extended | 8 | VCX | _Zero
ABSSetPrim | 2050 | Extended | 8 | VCX | _Zero
DriverControlsPrim | 2051 | Extended | 8 | VCX | _Zero
GetECMEngineData | 2055 | Extended | 8 | MGen | _Exportable
TranmsmissionSetPrim | 2064 | Extended | 8 | VCX | _Zero
TranmsmissionSetSec | 2065 | Extended | 8 | VCX | _Zero
VehicleSetPrim | 2066 | Extended | 8 | VCX | _Zero
GetVehicleDataA | 2067 | Extended | 8 | MGen
GetVehicleDataB | 2068 | Extended | 8 | MGen
GetVehicleDataC | 2069 | Extended | 8 | MGen
OpenLoopOutputsA | 2070 | Extended | 8 | VCX | _Zero
Bit Position
7 6 5 4 3 2 1 0
Byte Number
0 1 2 3 4 5 6 7

  - We can also use “macros”, an add2 extension supported by the CAN Database engine to provide additional facilities, such as controlling which parts of the database are exportable, hidden etc
  - Linking GUI user controls to parameters (Right-hand tab in tree)

DTB Vehicle Model – Further **model areas**

- A look at the model, covering the following areas
  - Model Configuration
    - MICROGen Initialisation Block
    - VISUALCONNX CCP Block (Used for reprogramming)
    - Set Model Timing
    - MPC555 Monitor
    - MPC555 QSPI Configuration
  - Version String
    - Try to update this for every build change
    - Make clear who the model developer is
    - Maintain source control (Use Doc block or a management tool)

DTB Vehicle Model – **Building the Model**

- How the build process takes place…
  - Make sure the model was loaded via “runme”
  - Remember to turn off “Test Control” in inputs sub-system.
  - Run model off-line first (Sometimes necessary)
  - Control-**B** to **build** at top level
  - Watch code, build and resolve issues as required
  - Output creates an “s19” extension file
    - This is a Motorola hex file that VISUALCONNX can download
    - This file contains the program for internal flash memory

Demo

ECHO is off.
Building bin/DIAB/555/internal_flash.dld
C:/MATLAB/R2007b/sys/perl/win32/bin/perl.exe -IC:/MATLAB/R2007b/toolbox/rtw/targets/mpc555dk/bin/win32
"LIBS = C:/MATLAB/R2007b/toolbox/rtw/targets/mpc555dk/drivers/lib/DIAB/lib_cmf_flash.555.a C:/MATLAB/
"SHARED_LIB = ../slprj/mpc555rt/_sharedutils/rtwshared.a"
"SHARED_LIB_TARGET = ../slprj/mpc555rt/_sharedutils/DIAB/555/rtwshared.a"
-- Compiling S-Record : ../Vehicle_Model_flash.s19 --- bin/DIAB/555/exceptions.o bin/DIAB/555/dec_han
--- LM_LICENSE_FILE="c:\flexlm\License.dat" ---
c:/diab/4.3g/Win32/bin/dcc -DMPC555_VARIANT -XO -Xlicense-wait -tPPC555EH:simple -D ROM_VERSION
c:/diab/4.3g/Win32/bin/ddump -R -v -m3 -w32 ../Vehicle_Model_flash.elf -o ../Vehicle_Model_flash.s19
C:/MATLAB/R2007b/sys/perl/win32/bin/perl.exe -IC:/MATLAB/R2007b/toolbox/rtw/targets/mpc555dk/bin/win32
-- DIAB build for ../Vehicle_Model_flash complete
-- ../Vehicle_Model_flash.elf ready for download
-- ../Vehicle_Model_flash.s19 ready for download
\#\#\# Completed MPC5xx make phase
Successful completion of MPC Real-Time Target build procedure for model: Vehicle_Model
>>

DTB Vehicle Model – **Learning the MICROGen blockset**

- This may require to be a separate training session
  - Some I/O blocks may not require changing for the DTB
  - Can show the full facilities available or a sub-set
    - Front Panel
      - LCD Text, String Operations, LEDs, Sounder, MPC55 Monitor
    - VISUALCONNX
      - Only used for CCP download for DTB (CCP communications not used for GUI)
    - Inputs
      - Genix, Standard, Ignition State (Relates to MICROGen Ignition **not** DTB!)
    - Outputs
      - Analogue Outputs, Aux Outputs, PWM Outputs, Relay Outputs, Pulse Stream Repeater, Synchronised Waveform Repeater
    - Additional CAN
      - CAN C and CAN D
    - RS232
      - Serial Communications, Serial COMMS BI
    - Power Control
      - Not relevant to DTB

DTB Vehicle Model – **Are MICROGen blocks all used?**

- Front Panel
  - LCD Text, String Operations, LEDs and MPC55 Monitor are all used in the DTB model
  - Sounder is there if required
- VISUALCONNX
  - Must be in the DTB model if reflash is to run without a power cycle
- Inputs
  - Genix and Standard both used in DTB model, Ignition State not required for DTB as power up all the time.
- Outputs
  - Analogue Outputs, Aux Outputs, PWM Outputs, Relay Outputs all used in the DTB model
  - Pulse Stream Repeater, Synchronised Waveform Repeater available for more complex wheelspeed for examlpe
- Additional CAN
  - CAN C and CAN D – used for reading off medium speed CAN in the future etc
- RS232
  - Serial Communications, Serial COMMS BI – Neither currently used, but may be required for PSU control, in the future (unless CAN contolled PSU is used)
- Power Control
  - Not relevant to DTB

Full Blocks Demo (Time Permitting)

DTB – Any question relating to MICROGen model

If any one has any questions please ask now.

Wrap up session

Feedback on session

Thank you.